PASSED

Computational Statistics - Group6 Lab1

Araya Eamrurksiri and Oscar Pettersson

February 1, 2016

Assignment1: Be careful with '=='

A pupil of a school is bad in arithmetic but good in programming. He writes a program to check if 1/3-1/4==1/12:

```
x1<-1/3;
x2<-1/4;
if (x1-x2==1/12){
print("Teacher said true")
} else{
print("Teacher lied")}
```

1.1 Check the result of this program. Comment why this happened.

```
print(x1 - x2, digits = 22)
```

```
## [1] 0.0833333333333333314829620
```

```
print(1 / 12, digits = 22)
```

```
## [1] 0.0833333333333333287074
```

So, both approximations of $\frac{1}{12}$ differ from the true value $0.8\bar{3}$ but R:s representation of $\frac{1}{12}$ differs the least. The representation for $\frac{1}{3}$ errs too, though not the one for $\frac{1}{4}$. Not because it has a finite number of decimals but because it is an exact power of 2:

```
print(x1, digits = 22) #1/3
```

```
## [1] 0.3333333333333333148296
```

```
print(x2, digits = 22) #1/4
```

```
## [1] 0.25
```

The error in $\frac{1}{3}$ causes the calculation of $x_1 - x_2$ to err but when this value is rounded to a value that is possible to store, the error gets even bigger. So, R:s representation of $\frac{1}{3}$ or $\frac{1}{12}$ contains one roundoff error each, while the representation of $x_1 - x_2$ contains the result of two cumulative roundoff errors.

1.2 Specify how the program can be modified to give a correct result

We used funtion `all.equal()` in R to help solve this problem. This function will test the two objects whether they are nearly equal or not and allow some error within the tolerance, $1.490116 * 10^{-8}$.

```
x1<-1/3;
x2<-1/4;
if (all.equal(x1-x2,1/12)){
    print("Teacher said true")
} else{
    print("Teacher lied")}
```

```
## [1] "Teacher said true"
```

Assignment2: Derivative

A widely known way to compute the derivative of function $f(x)$ in point x is to use

$$f'^{(x)} = \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

2.1 Write your own function computing the derivative of function $f(x) = x$ in this way. Take $\epsilon = 10^{-15}$

```
epsilon <- 10^-15
derivative <- function(x){
    result <- ( (x+epsilon) - x )/epsilon
    return(result)
}
```

2.2 Compute your derivative function at point x=100000

```
derivative(x=100000)
```

```
## [1] 0
```

2.3 What is the value you obtained? What is the real value of the derivative? Explain the reason behind the discovered difference

- Value obtained from the function: 0
- Real value of the derivative: 1

```
x <- 10000
(x+epsilon) - x
```

```
## [1] 0
```

As a matter of fact, the answer to the equation above shoul be equal to the value of epsilon, or 10^{-15}. However, the value which we get from R is 0. This is due to the fact that the significant digit of epsilon is lost due to the floating point.

Assignment3: Variance

A known formula for estimaing variance is

$$Var(x) = \frac{1}{n-1}(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2)$$

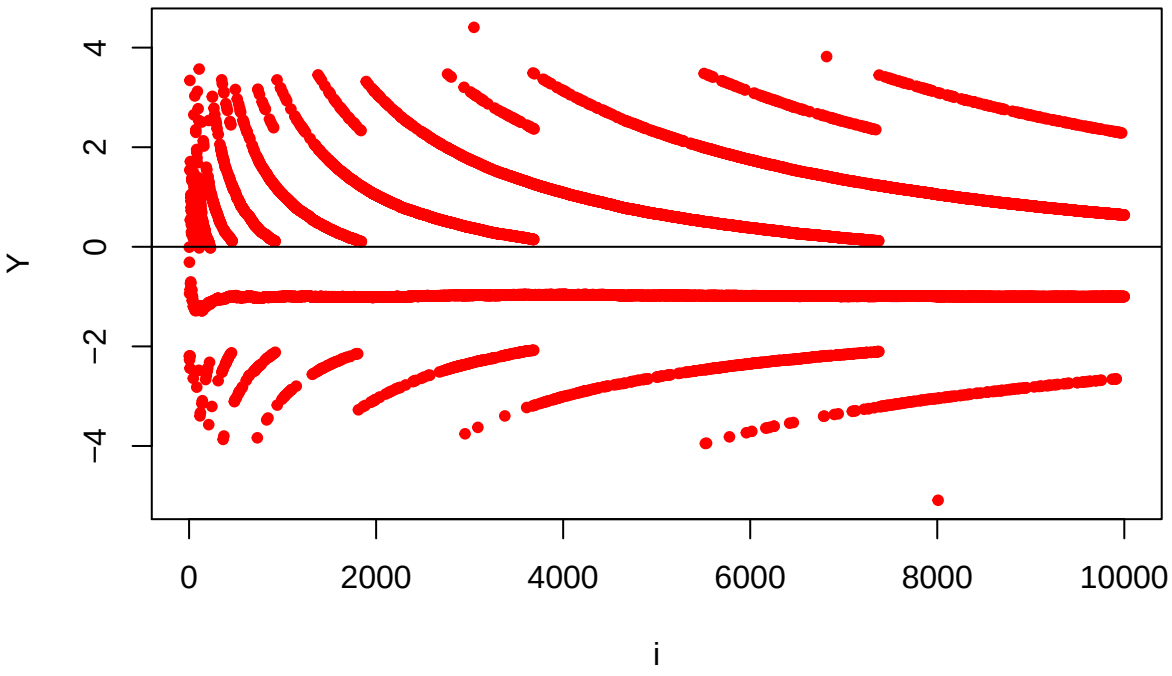
3.1 Write your own function `myvar` estimating variance in this way

```
myvar <- function(x){
  n <- length(x)
  Var <- (1 / (n - 1)) * (sum(x ^ 2) - (1 / n) * (sum(x)) ^ 2)
  return(Var)
}
```

3.2 Generate vector $x = (x_1 ... x_{10000})$ with 10000 random numbers, normally distributed with mean 10^8 and variance 1

```
n <- 10000
set.seed(12345)
x <- rnorm(n = n, mean = 10 ^ 8, sd = 1)
```

3.3 For each subset $X_i = x_1 ... x_i, i = 1 ... 10000$ compute difference $Y_i = myvar(X_i) - var(X_i)$, where $var(X_i)$ is a standard variance estimation function in R. Plot the dependence Y_i on i. Draw necessary conclusions. How well does your function work? What is the reason behind such behavior?



```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## 0.00768 0.97910 0.99240 0.99390 1.00000 1.30700       1
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -4.0920  0.0000  1.2190  0.9723  1.8940  5.3810       1
```

It seems that the function `myvar` is not really working well. The expected result should have value center around 0; however, we can see that there is a straight line around -1 and different kind of values.

The reason why function `myvar` does not work properly even though the formula is correct is because of the catastrophic cancellation. When dealing with large values, these two terms, $\sum_{i=1}^{n} x_i^2$ and $(\sum_{i=1}^{n} x_i)^2$, will become nearly equal to each other.

Assignment4: Linear Algebra

4.1 Import the data `tecator.csv` to R

4.2 Optimal regression coefficients can be found by solving a system of the type $A\beta = b$ where $A = X^T X$ and $b = X^T Y$. Compute A and b for the given data set.

```
X <- cbind(1, as.matrix(tecator[, c(2:102, 104)]))
Y <- as.matrix(tecator[, 103, drop = FALSE])
A <- t(X) %*% X
b <- t(X) %*% Y
```

4.3 Try to solve $A\beta = b$ with default solver `solve()`. What kind of result did you get? How can this result be explained?

```
beta <- solve(a = A, b = b)
```

```
## Error in solve.default(a = A, b = b): system is computationally singular: reciprocal condition numbe
```

An error of using this function is displayed above. The error states that the system is computationally singular meaning that the design matrix, in this case matrix A, is not invertible or a singular matrix. This is because of the high correlation and rounding errors. The determinant of matrix A is so small that leading R to an assumption that it is equal to 0.

4.4 Check the condition number of the matrix A and consider how it is related to your conclusion in step 3.

We use `kappa()` function in R to help find the condition number which is shown below.

```
## [1] 8.523517e+14
```

The number is pretty high which means we have no indication that the system of equation is well behaved; however, we can't also guarantee that it's always going to be bad. In this case, about 14 decimals of precition have been lost for solving this equation. This is also consistent with the result we have got from step 3. Even though we can compute the value of β, we wouldn't recommend to use it for further analysis as the result might not be correct indicated by the large conditional number.

4.5 Scale the data set and repeat steps 2-4. How the result has changed and why?

```
X_new <- X
X_new[, -1] <- scale(x = X[, -1], center = TRUE, scale = TRUE)
A_new <- t(X_new) %*% X_new
b_new <- t(X_new) %*% Y
beta_new <- solve(a = A_new, b = b_new)
```

This time we could finally get the result of β (didn't show in this report). The only different thing that have been changed so far in this step is scaling the input data in the beginning. After examining the variables in dataset, we see that the value of `Moisture` have quite a bigger magnitude than the rest of variables. Therefore, it affects the last column of matrix A.

Scaling data makes the condition number change its value as can be seen below. However, it does not make a condition of system to get any better.

```
## [1] 490471520661
```

R-code

```
## x1<-1/3;
## x2<-1/4;
## if (x1-x2==1/12){
## print("Teacher said true")
## } else{
## print("Teacher lied")}
x1<-1/3
x2<-1/4
print(x1 - x2, digits = 22)
print(1 / 12, digits = 22)
print(x1, digits = 22) #1/3
print(x2, digits = 22) #1/4
x1<-1/3;
x2<-1/4;
if (all.equal(x1-x2,1/12)){
    print("Teacher said true")
} else{
    print("Teacher lied")}
epsilon <- 10^-15
derivative <- function(x){
    result <- ( (x+epsilon) - x )/epsilon
    return(result)
}
derivative(x=100000)
x <- 10000
(x+epsilon) - x
myvar <- function(x){
  n <- length(x)
  Var <- (1 / (n - 1)) * (sum(x ^ 2) - (1 / n) * (sum(x)) ^ 2)
  return(Var)
```

```
}
n <- 10000
set.seed(12345)
x <- rnorm(n = n, mean = 10 ^ 8, sd = 1)
var_list <- myvar_list <- rep(NA_real_, n)
for(i in 1:n){
  var_list[i] <- var(x[1:i])
  myvar_list[i] <- myvar(x[1:i])
}
Y <- myvar_list - var_list
plot(Y ~ c(1:n), pch = 20, col = "red", xlab = "i")
abline(h = 0)
summary(var_list)
summary(myvar_list)
tecator <- read.csv("/Users/lynn/Documents/LiU/732A38 Computational statistics/Lab1/tecator.csv")
X <- cbind(1, as.matrix(tecator[, c(2:102, 104)]))
Y <- as.matrix(tecator[, 103, drop = FALSE])
A <- t(X) %*% X
b <- t(X) %*% Y
beta <- solve(a = A, b = b)
kappa(A)
X_new <- X
X_new[, -1] <- scale(x = X[, -1], center = TRUE, scale = TRUE)
A_new <- t(X_new) %*% X_new
b_new <- t(X_new) %*% Y
beta_new <- solve(a = A_new, b = b_new)
kappa(A_new)
## NA
```